



Policy Enforcer API Developer Guide



Modified: 2018-07-11

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. and/or its affiliates in the United States and other countries. All other trademarks may be property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Policy Enforcer API Developer Guide

Copyright © 2018 Juniper Networks, Inc. All rights reserved.

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at <https://www.juniper.net/support/eula/>. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

Chapter 1	Configuring and Using Policy Controller API	5
	Understanding Policy Controller API	5
	Configuring Policy Controller API	6
	Using Policy Controller API	8
	Site API Usage Examples	8
	Policy Enforcement Group API Usage Examples	10
	Threat Policy API Usage Examples	12
	Custom Feed API Usage Examples	14
	Geo IP API Usage Examples	18
	Log API Usage Examples	19

CHAPTER 1

Configuring and Using Policy Controller API

- [Understanding Policy Controller API on page 5](#)
- [Configuring Policy Controller API on page 6](#)
- [Using Policy Controller API on page 8](#)

Understanding Policy Controller API

The Policy Controller API provides advanced threat prevention policies that you can apply on your network security devices and monitor them for attacks. You can use this API to gather and aggregate threat information from multiple locations and devices, both physical and virtual, as well as from third-party solutions. You can use this information to assess and manage threats on your network.

The Policy Controller API is a pivotal component of Policy Enforcer. Policy Enforcer provides centralized, integrated threat management of all your security devices (both physical and virtual). For more information on Policy Enforcer, see [Policy Enforcer Administration Guide](#).

The following is a list of functionalities that you can perform using the various APIs defined under the Policy Controller API:

- Configure Policy Enforcer settings using the Config API.
- Create secure fabrics/sites using the Site API.
- Create feed sources/Sky ATP realms using the Feed Source API.
- Create policy enforcement groups using the Policy Enforcement Group API.
- Create threat prevention policies using the Threat Policy API.
- Create custom feeds using the Custom Feed API.
- Create Geo IP policies using the Geo IP API.
- Retrieve all log files in zip format using the Logs API.

You can also perform these activities using the Policy Enforcer UI. For more information, see [Policy Enforcer Administration Guide](#).

- Related Documentation**
- [Configuring Policy Controller API on page 6](#)
 - [Using Policy Controller API on page 8](#)
 - [Policy Enforcer API Reference Guide](#)

Configuring Policy Controller API

You must perform some initial configuration and setup activities through the Policy Enforcer UI before you begin using the Policy Controller API.

1. Log into the Policy Enforcer UI and configure the threat prevention type. You can configure Policy Enforcer to run in the following modes based on the threat prevention type you select:

- Cloud only or cloud feeds mode
- Sky ATP mode
- Sky ATP with Policy Enforcer mode

For more information on configuring threat prevention types, see [Policy Enforcer Administration Guide](#).

2. Invoke the Config API with HTTP basic authentication using your Policy Enforcer server SSH user credentials, as shown in the following example:

```
POST <context>/api/v1/controller/configs
Content-Type: application/json
Authorization: Basic <base 64 encoded
(sssh username of Policy Enforcer>:<ssh password of Policy
Enforcer>)>

{"configs": {
  "sdsn": true, "cloudOnly": false,
  "ems": {"url": <URL of Security Director>, "username": "Policy
Enforcer_user",
    "password": <ssh password of Policy Enforcer>},
  "restApi": {"username": <REST API username>,
    "password": <REST API password>}
}
```

**NOTE:**

- You must use only https for the ems URL.
- Ensure that you are using the correct Policy Enforcer username and password that successfully authenticate Security Director.
- The REST API username and password are the new credentials for the REST APIs.

Based on the mode you have selected in the Policy Enforcer UI, you can specify the configuration using the Config API as follows:

- a. **Cloud only or cloud feeds mode**—You can configure Policy Enforcer in the cloud only or cloud feeds mode as follows:

```
"configs": {
  "sdsn": false
  "cloudonly": true
}
```

- b. **Sky ATP mode**—You can configure Policy Enforcer in the Sky ATP mode as follows:

```
"configs": {
  "sdsn": false
  "cloudonly": false
}
```

- c. **Sky ATP with Policy Enforcer mode**—You can configure Policy Enforcer in the Sky ATP with Policy Enforcer mode as follows:

```
"configs": {
  "sdsn": true
  "cloudonly": false
}
```



NOTE: Ensure that the values of `sdsn` and `cloudonly` reflect the mode you have selected in the Policy Enforcer UI.

3. You can also use your REST API user credentials for HTTP basic authentication to make any Policy Enforcer REST API calls. To do so, you must first create your REST API username and password. You can use any value as the username and password, for example, `admin/admin` or `abcd/wxyz`.

**Related
Documentation**

- [Understanding Policy Controller API on page 5](#)
- [Using Policy Controller API on page 8](#)
- [Policy Enforcer API Reference Guide](#)

Using Policy Controller API

The following sections provide usage examples for the various APIs defined in the Policy Controller API:



NOTE: For usage examples of the Config API, see [“Configuring Policy Controller API”](#) on page 6.

- [Site API Usage Examples on page 8](#)
- [Policy Enforcement Group API Usage Examples on page 10](#)
- [Threat Policy API Usage Examples on page 12](#)
- [Custom Feed API Usage Examples on page 14](#)
- [Geo IP API Usage Examples on page 18](#)
- [Log API Usage Examples on page 19](#)

Site API Usage Examples

The following are usage examples for the Site API:

Usage Example 1 - Creating a site

POST <context>/api/v1/controller/sites
Content-Type: application/json
STATUS: 200

```
"site": {
  "name": "sunnyvale",
  "domain": "SD domain name",
  "description": "location sunnyvale",
  "feedSourceId": null,
  "devices": [{"name": "dev1", "emsId": "1234", "serialNumber": "ABCD1234",
    "ip": "192.0.2.0", "model": "srx1500",
    "role": "EDGE_FIREWALL"},
    {"name": "dev2", "emsId": "1235", "serialNumber": "ABCD1235",
    "ip": "192.0.2.1", "model": "srx550",
    "role": "AGGREGATION_FIREWALL"},
    {"name": "dev3", "emsId": "1236", "serialNumber": "ABCD1236",
    "ip": "192.0.2.1", "model": "srx550",
    "role": "EDGE_FIREWALL"},
    {"name": "dev4", "emsId": "1237", "serialNumber": "ABCD1237",
    "ip": "192.0.2.2", "model": "ex4500",
    "role": "CORE_SWITCH"},
    {"name": "dev5", "emsId": "1238", "serialNumber": "ABCD1238",
    "ip": "192.0.2.3", "model": "ex4300",
    "role": "ACCESS_SWITCH"}
  ]
}
```

where:

- **emsId** is the device identifier from Security Director. You can use the Space REST API to obtain the device identifier using [Device Management API](#).

- **feedSourceId** is the skyATP realm and its value is **NULL** during the POST operation. When a site is associated to realms, you can update the site with **feedSourceId**.
- The value of **enrollStatus** can be **ENROLL_SUCCESS**, **ENROLL_FAILURE**, **DISENROLL_SUCCESS**, or **DISENROLL_FAILURE**.

Usage Example 2 - Retrieving a site

```
GET <context>/api/v1/controller/sites/{siteId}
Content-Type: application/json
STATUS: 200

"site": {
  "id": "uuid-1234"
  "uri": "https://<host>/<context>/api/v1/controller/sites/uuid-1234",
  "name": "sunnyvale",
  "domain": "SD domain name",
  "description": "location sunnyvale",
  "feedSourceId": "uuid-1234",
  "devices": [{
    "deviceId": "11111", "name": "dev1", "emsId": "1234",
    "serialNumber": "ABCD1234", "ip": "192.0.2.0",
    "model": "srx1500", "role": "EDGE_FIREWALL",
    "enrollStatus": "ENROLL_SUCCESS"},
    {
    "deviceId": "22222", "name": "dev2", "emsId": "1235",
    "serialNumber": "ABCD1234", "ip": "192.0.2.1",
    "model": "srx550", "role": "AGGREGATION_FIREWALL"},
    {
    "deviceId": "33333", "name": "dev3", "emsId": "1236",
    "serialNumber": "ABCD1234", "ip": "192.0.2.1",
    "model": "srx550", "role": "EDGE_FIREWALL",
    "enrollStatus": "ENROLL_FAILURE",
    "statusReason": "failed .."},
    {
    "deviceId": "44444", "name": "dev4", "emsId": "1237",
    "serialNumber": "ABCD1234", "ip": "192.0.2.2",
    "model": "ex4500", "role": "CORE_SWITCH"},
    {
    "deviceId": "55555", "name": "dev5", "emsId": "1238",
    "serialNumber": "ABCD1234", "ip": "192.0.2.3",
    "model": "ex4300", "role": "ACCESS_SWITCH"}
  ]
}
```

Usage Example 3 - Updating a site based on siteId

```
PUT <context>/api/v1/controller/sites/{siteId}
Content-Type: application/json
STATUS: 200

"site": {
  "name": "sunnyvale",
  "domain": "SD domain name",
  "description": "location sunnyvale",
  "feedSourceId": "uuid-site-1234",
  "devices": [{
    "name": "dev1", "emsId": "1234", "serialNumber": "ABCD1234",
    "ip": "192.0.2.0", "model": "srx1500",
    "role": "EDGE_FIREWALL"},
    {
    "name": "dev2", "emsId": "1235", "serialNumber": "ABCD1235",
    "ip": "192.0.2.1", "model": "srx550",
    "role": "AGGREGATION_FIREWALL"},
    {
    "name": "dev3", "emsId": "1236", "serialNumber": "ABCD1236",
    "ip": "192.0.2.1", "model": "srx550",
```

```

        "role": "EDGE_FIREWALL"},
{"name": "dev4", "emsId": "1237", "serialNumber": "ABCD1237",

        "ip": "192.0.2.2", "model": "ex4500",
        "role": "CORE_SWITCH"},
{"name": "dev5", "emsId": "1238", "serialNumber": "ABCD1238",

        "ip": "192.0.2.3", "model": "ex4300",
        "role": "ACCESS_SWITCH"}
    ]
}

```

**Usage Example 4 -
Retrieving the updated
site to check if the
updates are present**

GET <context>/api/v1/controller/sites
Content-Type: application/json
STATUS: 200

```

"sites": {
  "uri": "https://<host>/<context>/api/v1/controller/sites",
  "total": 2,
  "site": [
    {"id": "uuid-1234",
      "uri": "https://<host>/<context>/api/v1/controller/sites/uuid-1234",
      "name": "sunnyvale", "domain": "SD domain name",
      "description": "location sunnyvale", "feedSourceId": "uuid-1234",
      "devices": [{"deviceId": "11111", "name": "dev1", "ip", "192.0.2.0"},
        {"deviceId": "22222", "name": "dev2", "ip", "192.0.2.1"}]},
    {"id": "uuid-1235",
      "uri": "https://<host>/<context>/api/v1/controller/sites/uuid-1235",
      "name": "Westford", "domain": "SD domain name",
      "description": "location westford", "feedSourceId": "uuid-1234",
      "devices": [{"deviceId": "33333", "name": "dev3", "ip", "192.0.2.1"},
        {"deviceId": "44444", "name": "dev4", "ip", "192.0.2.2"}]}
  ]
}

```

**Usage Example 5 -
Deleting a site**

DELETE <context>/api/v1/controller/sites/{siteId}
STATUS: 204

Policy Enforcement Group API Usage Examples

The following are usage examples for the Policy Enforcement Group API:

**Usage Example 1 -
Creating a new Policy
Enforcement Group**

POST <context>/api/v1/controller/policyGroups
Content-Type: application/json
STATUS: 200

```

"policyGroup": {
  "name": "sunnyvale",
  "domain": "SD domain name",
  "feedSourceId": "uuid-realm-1234",
  "description": "sunnyvale user endpoints",
  "groupType": "IP",
  "sites": [{"siteId": "uuid-111", "name": "bldg-A",
    "uri", "/api/v1/controller/Sites/uuid-111"},
    {"siteId": "uuid-222", "name": "bldg-B",
    "uri", "/api/v1/controller/Sites/uuid-222"},

```

```

        {"siteId": "uuid-333", "name": "bldg-6",
         "uri", "/api/v1/controller/Sites/uuid-333"}
      ],
      "addressGroups": ["192.0.2.0/24", "198.51.100.0-198.51.100.255",
                        "203.0.113.0"]
    }

```

where:

- **sites** and **addressGroups** are mutually exclusive.
- The value of **addressGroups** can be a single IP, an IP range, or an IP subnet.
- If the value of **groupType** is **IP**, **addressGroups** are populated; if the value is **LOCATION**, sites are populated.

Usage Example 2 - Retrieving a specific policy enforcement group based on policyGroupId

```

GET <context>/api/v1/controller/policyGroups/{policyGroupId}
Content-Type: application/json
STATUS: 200

"policyGroup": {
  "id": "uuid-1234",
  "uri": "https://<host>/<context>/api/v1/controller/policyGroups/uuid-1234",

  "name": "sunnyvale",
  "domain": "SD domain name",
  "feedSourceId": "uuid-realm-1234",
  "description": "sunnyvale user endpoints",
  "sites": [],
  "addressGroups": ["192.0.2.0/24", "198.51.100.0-198.51.100.255",
                    "203.0.113.0"]
}

```

Usage Example 3 - Updating a specific policy enforcement group based on policyGroupId

```

PUT <context>/api/v1/controller/policyGroups/{policyGroupId}
Content-Type: application/json
STATUS: 200

"policyGroup": {
  "name": "sunnyvale",
  "domain": "SD domain name",
  "feedSourceId": "uuid-realm-1234",
  "description": "sunnyvale user endpoints",
  "sites": [],
  "addressGroups": ["192.0.2.0/24", "198.51.100.0-198.51.100.255",
                    "203.0.113.0"]
}

```

Usage Example 4 - Retrieving the updated policy enforcement group to check if the updates are present

```

GET <context>/api/v1/controller/policyGroups
Content-Type: application/json
STATUS: 200

"policyGroups": {
  "uri": "https://<host>/<context>/api/v1/controller/policyGroups",
  "total": 2,
  "policyGroup": [
    {"id": "uuid-1234",

```

```

"uri":"https://<host>/<context>/api/v1/controller/policyGroups/uuid-1234",

"name": "sunnyvale", "domain": "SD domain name",
"description": "sunnyvale user endpoints", "feedSourceId",
"uuid-realm-1234"
"sites": [],
"addressGroups":
["192.0.2.0/24", "198.51.100.0-198.51.100.255", "203.0.113.0"]},
{"id": "uuid-1234",
"uri":"https://<host>/<context>/api/v1/controller/policyGroups/uuid-1234",

"name": "sunnyvale", "domain": "SD domain name",
"description": "sunnyvale user endpoints", "feedSourceId",
"uuid-realm-1235"
"sites": [{"siteId": "uuid-111", "name": "bldg-A", "uri",
"/api/v1/controller/Sites/uuid-111"},
{"siteId": "uuid-222", "name": "bldg-B", "uri",
"/api/v1/controller/Sites/uuid-222"},
{"siteId": "uuid-333", "name": "bldg-6", "uri",
"/api/v1/controller/Sites/uuid-333"}
],
"addressGroups": []},
]
}

```

**Usage Example 5 -
Deleting a policy
enforcement group**

```

DELETE <context>/api/v1/controller/policyGroups/{policyGroupId}
STATUS: 204

```

Threat Policy API Usage Examples

The following are usage examples for the Threat Policy API:

**Usage Example 1 -
Creating a new Threat
Policy**

```

POST <context>/api/v1/controller/threatPolicys
Content-Type: application/json
STATUS: 200

```

```

"threatPolicy": {
  "name": "simplePolicy",
  "domain": "SD domain name",
  "description": "with all profiles",
  "profiles": [{
    "feedType": "CnC",
    "actions": [{"threatLevelStart": "0", "threatLevelEnd": "4",
      "action": "PERMIT"},
      {"threatLevelStart": "5", "threatLevelEnd": "7",
      "action": "LOG"},
      {"threatLevelStart": "8", "threatLevelEnd": "9",
      "action": "BLOCK_CLOSE", "redirectUrl": "",
      "customMessage": ""}]
    }, {
    "feedType": "INFECTED_HOST",
    "actions": [{"threatLevelStart": "0", "threatLevelEnd": "4",
      "action": "PERMIT"},
      {"threatLevelStart": "8", "threatLevelEnd": "9",
      "action": "BLOCK_QUARANTINE",
      "quarantineVlanId": "911"}]
  }, {

```

```

    "feedType": "MALWARE", "malwareProfileName": "scanAll",
    "https": true, "actions": [{"threatLevelStart": "0",
        "threatLevelEnd": "6", "action": "PERMIT"},
        {"threatLevelStart": "7", "threatLevelEnd": "9",
        "action": "BLOCK_CLOSE", "redirectUrl": "",
        "customMessage": "call IT support"}]
    }, {
    "feedType": "SMTP", "attachmentProfileName": "scanAll",
    "actions": [{"threatLevelStart": "0", "threatLevelEnd": "6",
        "action": "PERMIT"},
        {"threatLevelStart": "7", "threatLevelEnd": "9",
        "action": "BLOCK_DROP"}]
    }
  ],
  "secondaryActions": ["LOG"],
  "policyGroups": [{"policyGroupId": "uu-123", "name": "peg1"},
    {"policyGroupId": "uu-456", "name": "peg2"}],
  "deployStatus": "DRAFT"
}

```

where:

- The value of *action* can be **PERMIT**, **LOG**, **BLOCK_DROP**, **BLOCK_CLOSE**, or **BLOCK_QUARANTINE**.
- The value of *secondaryAction* can be **LOG_ALL**, **LOG_BLOCKED**, or **NONE**.
- If you specify **MALWARE** as the *feedType*, SRX takes a single threat level threshold value, that is, it allows two actions — permit and block.
- If you specify **GEO_IP** as the *feedType*, then the SRX Series device has no threshold and allows permit or block.
- For *deployStatus*, you do not have to specify the values **DRAFT**, **ANALYSIS_PROGRESS**, **READY_TO_DEPLOY**, and **DEPLOYED** for POST and PUT operations.

Usage Example 2 - Updating a threat policy

```

PUT <context>/api/v1/controller/threatPolicys/uuid-1234/emsData
Content-Type: application/json
STATUS: 200
STATUS: 500 (It can have following errors)
    "no PerimeterFirewall found based on PEG, skipping analysis"
    "ATP analysis policy: <xyz> has aamw/infected-host profile, no argon
capable device, skipping analysis"

```

```

"threatPolicy": {
  "name": "simplePolicy",
  "domain": "SD domain name",
  "description": "with all profiles",
  "profiles": [],
  "secondaryActions": ["LOG"],
  "policyGroups": [{"policyGroupId": "uu-123", "name": "peg1"},
    {"policyGroupId": "uu-456", "name": "peg2"}],
  "deployStatus": "DRAFT",
  "emsAnalysisId": "uuid-policy-analysis",
  "emsPublishUpdateId": "publish-update-job-id"
}

```

Usage Example 3 - Retrieving a specific threat policy based on threatPolicyId

GET <context>/api/v1/controller/threatPolicys/uuid-1234
Content-Type: application/json
STATUS: 200

```
"threatPolicy": {
  "id": "uuid-1234",
  "uri": "https://<host>/<context>/api/v1/controller/threatPolicys/uuid-1234",

  "name": "simplePolicy",
  "domain": "SD domain name",
  "description": "with all profiles",
  "profiles": [{
    "feedType": "CnC",
    "actions": [{
      "threatLevelStart": "0", "threatLevelEnd": "4",
      "action": "PERMIT"},
      {
        "threatLevelStart": "5", "threatLevelEnd": "7",
        "action": "LOG"},
      {
        "threatLevelStart": "8", "threatLevelEnd": "9",
        "action": "BLOCK_CLOSE",
        "redirectUrl": "", "customMessage": ""}
    ], {
      "feedType": "INFECTED_HOST",
      "actions": [{
        "threatLevelStart": "0", "threatLevelEnd": "4",
        "action": "PERMIT"},
        {
          "threatLevelStart": "8", "threatLevelEnd": "9",
          "action": "BLOCK_QUARANTINE",
          "quarantineVlanName": "911"}
      ], {
        "feedType": "MALWARE", "malwareProfileName": "scanAll",
        "https": true, "actions": [{
          "threatLevelStart": "0",
          "threatLevelEnd": "6", "action": "PERMIT"},
          {
            "threatLevelStart": "7", "threatLevelEnd": "9",
            "action": "BLOCK_CLOSE",
            "redirectUrl": "", "customMessage": "call IT support"}
        ], {
          "feedType": "SMTP", "attachmentProfileName": "scanAll",
          "actions": [{
            "threatLevelStart": "0", "threatLevelEnd": "6",
            "action": "PERMIT"},
            {
              "threatLevelStart": "7", "threatLevelEnd": "9",
              "action": "BLOCK_DROP"}
          ]
        },
        "secondaryActions": ["LOG"],
        "policyGroups": [{
          "policyGroupId": "uu-123", "name": "peg1"},
          {
            "policyGroupId": "uu-456", "name": "peg2"}],
        "deployStatus": "DRAFT",
        "deployDevices": [{
          "name": "device1", "deviceId": "uuid1234"}],
        "skipDevices": [{
          "name": "device2", "deviceId": "uuid5678"}]
      }
    ]
  }
}
```

Usage Example 4 - Deleting a threat policy

DELETE <context>/api/v1/controller/threatPolicys/uuid-1234,
STATUS: 204

Custom Feed API Usage Examples

The following are usage examples for the Custom Feed API:

Usage Example 1 - Creating a new CustomFeed

```
POST <context>/api/v1/controller/customFeeds/<feedType>/param/<inputType>/<name>
Content-Type: application/json
Accept: application/json
STATUS: 200
Body:
"customFeed": {
  "domain": "SD domain name",
  "description": "safe IPs",
  "content": ["192.0.2.0/24", "198.51.100.0-198.51.100.255"]
}
Response:
"customFeed": {
  "id": "uuid-1234",
  "emsVersion": 0,
  "createTs": 1479328662,
  "emsAddressId": null,
  "updateTs": null
  "uri": "/api/v1/controller/customFeeds/uuid-1234",
  "name": "customGoodIPs",
  "domain": "SD domain name",
  "description": "safe IPs",
  "feedType": "Whitelist",
  "inputType": "ip",
  "content": ["192.0.2.0/24", "198.51.100.0-198.51.100.255"]
}
```

where:

- The value of *feedType* can be **Blacklist**, **Whitelist**, or **Dynamic-Address**.
- The value of *content* can be a list of IP addresses, an IP range, or a subnet for a **Blacklist**, **Whitelist** and, **Dynamic-Address**.
- The value of *inputType* can be an IP, URL or a domain for a **Blacklist**, **Whitelist** and, **Dynamic-Address**.

Usage Example 2- Creating a new CustomFeed with Infected-Host feedtype

```
POST <context>/api/v1/controller/customFeeds/<feedType>/param/<inputType>/<name>
Content-Type: application/json
Accept: application/json
STATUS: 200
Body:
"customFeed": {
  "domain": "SD domain name",
  "description": "infected IPs",
  "content": {"add": ["192.0.2.0", "198.51.100.0"], {"delete":
["198.51.100.255"]}}
}
Response:
"customFeed": {
  "id": "uuid-1234",
  "emsVersion": 0,
  "createTs": 1479328662,
  "emsAddressId": null,
  "updateTs": null
  "uri": "/api/v1/controller/customFeeds/uuid-1234",
  "name": "customIPs",
  "domain": "SD domain name",
  "description": "infected IPs",
```

```
"feedType": "Infected-Hosts",
"inputType": "ip",
"content": {"add": ["192.0.2.0", "198.51.100.0"], {"delete":
["198.51.100.255"]}]
}
```

where:

- The value of *feedType* is **Infected-Hosts**.
- The value of *content* can be a list of IP addresses.
- The value of *inputType* can be and an IP address.

Usage Example 3- Retrieving a specific custom feed based on CustomFeed Id

```
GET <context>/api/v1/controller/customFeeds/<feedType>/param/<inputType>/<name>
Content-Type: application/json
STATUS: 200
Response:
"customFeed": {
  "id": "uuid-1234",
  "emsVersion":0,
  "createTs":1479328662,
  "emsAddressId":null,
  "updateTs":null
  "uri":"/api/v1/controller/customFeeds/uuid-1234",
  "name": "customGoodIPs",
  "domain": "SD domain name",
  "description": "safe IPs",
  "feedType": "Whitelist",
  "inputType": "ip",
  "content": ["192.0.2.0/24", "198.51.100.0-198.51.100.255"]
}
```

Usage Example 4 - Retrieving a specific infected-host custom feed based on CustomFeed Id

```
GET <context>/api/v1/controller/customFeeds/<feedType>/param/<inputType>/<name>
Content-Type: application/json
STATUS: 200
Response:
"customFeed": {
  "id": "uuid-1234",
  "emsVersion":0,
  "createTs":1479328662,
  "emsAddressId":null,
  "updateTs":null
  "uri":"/api/v1/controller/customFeeds/uuid-1234",
  "name": "customIPs",
  "domain": "SD domain name",
  "description": "infected IPs",
  "feedType": "Infected-Hosts",
  "inputType": "ip",
  "content": {"add": ["192.0.2.0", "198.51.100.0"],
{"delete": ["198.51.100.255"]}]
}
```

Usage Example 5- Retrieving the list of custom feeds

```
GET <context>/api/v1/controller/customFeeds//param/
Content-Type: application/json
STATUS: 200
```



```

Response:
"customFeeds": {
  "uri": "/api/v1/controller/customFeeds",
  "total": 2,
  "customFeed": [
    {
      "id": "uuid-1234", "emsVersion": 0, "createTs": 1479328662,
      "emsAddressId": null, "updateTs": null,
      "name": "customGoodIPs", "domain": "SD domain name",
      "description": "safe IPs", "feedType": "Whitelist",
      "content": ["192.0.2.0/24", "198.51.100.0-198.51.100.255"]},
    {
      "id": "uuid-3456", "emsVersion": 0, "createTs": 1479328662,
      "emsAddressId": null, "updateTs": null, "name": "customBadIPs",
      "domain": "SD domain name", "description": "bad IPs",
      "feedType": "Blacklist", "content": ["192.0.2.0/24", "203.0.113.0"]},
    {
      "id": "uuid-5678", "emsVersion": 0, "createTs": 1479328662,
      "emsAddressId": null, "updateTs": null, "name": "dynamicIPs",
      "domain": "SD domain name", "description": "misc IPs",
      "feedType": "Dynamic-Address", "content": ["192.0.2.0/24"]}
    {
      "id": "uuid-5678", "emsVersion": 0, "createTs": 1479328662,
      "emsAddressId": null, "updateTs": null, "name": "dynamicIPs",
      "domain": "SD domain name", "description": "infected IPs",
      "feedType": "Infected-Hosts",
      "content": {
        "add": ["192.0.2.0", "198.51.100.0"],
        "delete": ["198.51.100.255"]}
    }
  ]
}

```

Usage Example 6- Updating a custom feed

```

PUT <context>/api/v1/controller/customFeeds/<feedType>/param/<inputType>/<name>
Content-Type: application/json
STATUS: 200

```

```

Body:
"customFeed": {
  "domain": "SD domain name",
  "description": "safe IPs",
  "content": ["192.0.2.0/24", "198.51.100.0-198.51.100.255"]
}

```

```

Response:
"customFeed": {
  "id": "uuid-1234",
  "emsVersion": 0,
  "createTs": 1479328662,
  "emsAddressId": null,
  "updateTs": null
  "uri": "/api/v1/controller/customFeeds/uuid-1234",
  "name": "customGoodIPs",
  "domain": "SD domain name",
  "description": "safe IPs",
  "feedType": "Whitelist",
  "inputType": "ip",
  "content": ["198.51.100.0-198.51.100.255", "203.0.113.0"]
}

```

Usage Example 7- Deleting a custom feed

```

DELETE
<context>/api/v1/controller/customFeeds/<feedType>/param/<inputType>/<name>
STATUS: 204

```

Geo IP API Usage Examples

The following are usage examples for Geo IP API:

Usage Example 1 - Creating a new Geo IP

POST <context>/api/v1/controller/geoIps
Content-Type: application/json
STATUS: 200

```
"geoIp": {  
  "name": "asia",  
  "domain": "SD domain name",  
  "description": "all asia countries",  
  "countrys": [CN, IN],  
  "action": "BLOCK_INBOUND",  
  "secondaryAction": "LOG"  
}
```

where:

- The value of **action** can be **BLOCK_INBOUND**, **BLOCK_OUTBOUND**, or **BLOCK_BOTH**.
- The value of **secondaryAction** can be **LOG** or **NONE**.



NOTE: The values for **action** and **secondaryAction** are only needed for SDSN.

Usage Example 2 - Retrieving a specific Geo IP based on geoIpId

GET <context>/api/v1/controller/geoIps/{geoIpId}
Content-Type: application/json
STATUS: 200

```
"geoIp": {  
  "id": "uuid-1234",  
  "uri": "https://<host>/<context>/api/v1/controller/geoIps/uuid-1234",  
  "name": "asia",  
  "domain": "SD domain name",  
  "description": "all asia countries",  
  "countrys": [CN, IN],  
  "action": "INBOUND",  
  "secondaryAction": "LOG"  
}
```

Usage Example 3 - Retrieving the list of Geo IPs

GET <context>/api/v1/controller/geoIps
Content-Type: application/json
STATUS: 200

```
"geoIps": {  
  "uri": "https://<host>/<context>/api/v1/controller/geoips",  
  "total": 2,  
  "geoip": [  
    {"id": "uuid-1234", "name": "asia", "domain": "SD domain name",  
      "description": "all asia countries", "countrys": [CN, IN],  
      "action": "INBOUND", "secondaryAction": "LOG"},  
    {"id": "uuid-1235", "name": "north korea", "domain": "SD domain name",  
      "description": "some countries", "countrys": [KP],
```

```
        "action": "INBOUND", "secondaryAction": "LOG"}
    ]
}
```

**Usage Example 4 -
Deleting Geo IP**

DELETE <context>/api/v1/controller/geoIps/{geoIpId}
STATUS: 204

Log API Usage Examples

The following is a usage example for Log API:

**Usage Example -
Retrieving all log files
in zip format**

GET <context>/api/v1/controller/logs
STATUS: 200

**Related
Documentation**

- [Understanding Policy Controller API on page 5](#)
- [Configuring Policy Controller API on page 6](#)
- [Policy Enforcer API Reference Guide](#)

